Understanding SOA Migration Using a Conceptual Framework

Razavian, M.; Lago, P.

published in
Journal of Systems Integration
2010

Understanding SOA Migration Using a Conceptual Framework

Maryam Razavian and Patricia Lago
Department of Computer Science, VU University Amsterdam, the Netherlands
m.razavian@few.vu.nl, patricia@cs.vu.nl

Abstract: *Migration of legacy assets to SOA embodies a key software engineering challenge. Over a decade there has been an increasing interest in the approaches addressing SOA migration. These approaches mainly differ in `what is migrated' and `how the migration is performed'. Such differences aggravate achieving a general understanding of `what SOA migration entails'. We describe what such migration process entails and what distinct conceptual elements systematically define the process. Based on the comprising conceptual elements the framework which is considered as a basis for understanding and assessment of different approaches is proposed. Further, the role of the migration framework in positioning and assessing the existing methods, is discussed. Finally, the procedure for positioning and mapping of migration approaches on the framework is explained using two example migration processes.*

Keywords: SOA, Flight Booking System, Migration Strategy, Migration Framework

1. Introduction

One of the key features of the service oriented paradigm is to facilitate reuse of business functions provided by legacy systems. The main motivation behind the modernization of legacy systems to SOA is to achieve the advantages offered by SOA and still reuse the embedded functionalities in the legacy systems. Although some characteristics of the SOA paradigm, such as support of loosely coupled services and interoperability, make the service enabling of legacy systems look to be straightforward, it constitutes a key challenge of service design.

Recently, migration of legacy systems to SOA has caught a lot of attention in both research and industry, and a number of methods have been proposed. A vast body of work in this area addresses exposing legacy code as (web) services [1,2]. Typically, the focus of these works is limited to implementation aspects of migration which usually covers techniques to alter a segment of legacy code to web services. A second family of approaches aims for covering the whole migration process. These approaches are comprised of two main sub processes: top-down service development and bottom-up service extraction. Planning of migration is the main focus of some other group of approaches [3,4]. Basically the feasibility of migration process is assessed based on business drivers and characteristics of the legacy system.

These methods reflect different perspectives on SOA migration. They mainly differ in the way they provide solutions for two challenging problems of what can be migrated (i.e. the legacy elements) and how the migration is performed (i.e. the migration process). However, there is still little conceptual characterization of what the legacy to SOA migration process entails. As a result, a common understanding of the SOA migration is difficult to achieve. Moreover, the lack of study on the state of the art makes the understanding, comparison and analysis of existing methods especially difficult. To solve this problem, we need to establish a framework for SOA migration which facilitates achieving a general understanding on SOA migration process.

In earlier work [11] we proposed a framework, called SOA-MF that embraces a holistic illustration of the SOA migration process, along with the distinct conceptual elements involved in such a process. SOA-MF facilitates the representation of the SOA migration processes in a unified manner and therefore provides the basis for their comparison and analysis. As such, different migration approaches may be mapped to a portion of (or all) the framework. In this paper the procedure for positioning and mapping of migration approaches on the framework is explained using two example migration processes. In addition, we describe how such mappings facilitate achieving a general understanding of `what SOA migration entails', through comparison and categorization of migration approaches.

The reminder of the paper is organized as follows. Section 2 presents a running example, which aims for clarifying the role of the framework in understanding and positioning the migration methods. In

Section 3, the proposed SOA migration framework is described. Section 4, discusses the use of the proposed framework. Finally, Section 5 concludes the paper.

2. Running Example

To get a feel for how different approaches provide solutions for SOA migration, we present the following example. Let's consider the hypothetical problem of migrating a flight booking system to SOA. This system supports a number of business processes including "search flight catalogue", "set traveling preference", "book flight" and "bill customer". For this problem, we propose two different solutions based on two different SOA migration approaches including SMART [3] and the method proposed by Sneed [2]. Section 2.1 and 2.2 respectively discuss SMART and Sneed's approach to flight booking migration. Having the same problem domain facilitates better understanding of the commonalities and differences among the selected migration approaches. We will return to this example and use it to clarify different aspects of our proposed framework throughout the paper. For the sake of simplicity, some details of these two approaches are ignored.

2.1 SMART Approach to Migration of Flight Booking System

SMART is an approach to make initial decisions about the feasibility of reusing the legacy systems within an SOA environment, along with an understanding of costs and risks involved. By using interviews and questionnaires, this method gathers information about legacy components, the target SOA, and potential services. Regarding the migration of the flight booking system, SMART provides answers for following questions.

- Which services should be created?
- Which legacy components should be migrated to services?
- How this transformation should be performed?

The SMART migration process entails a number of activities, which are applied in this example and are discussed below. As the first step, SMART defines the primary candidate services from existing business processes. Here, the services which have clear inputs and outputs are considered as candidate services. The next activity of SMART, "describe existing capability", gathers descriptive characteristics such as name, function, and size of existing components. In addition, information regarding the target SOA environment is extracted within "describe target environment" activity. In this example, services should provide contracts embracing different security policies of customers billing information. The "gap analysis" activity, extracts the required changes to components code for exposing their functionality as services given the service requirements, the service inputs and outputs, as well as the characteristics and components of the target SOA environment. Based on this, an estimate of the effort required to make these changes are provided. Finally, the strategy, embracing the options for proceeding with the migration effort, along with their associated risk is devised. Table 1, illustrates the migration strategy for flight booking system.

Service	Migration method	Level of risk
get flight catalog	Wrap related functions in the Flight Catalog component	low
bill customer	Extract the billing functionality from Flight reservation component Create code for implementing different policies of billing information security	high
get flight information	Create an interface to Flight Catalog component	low
get customer information	Create an interface to Customer component	low

Table 1: Flight Booking System Migration Strategy

1.2 Sneed's Approach to Migration of Flight Booking System

The migration process proposed in [2] starts from identifying candidates for services by performing portfolio analysis and listing out essential business rules. Consider the "bill customer business process" embracing the secondary business rules including "aggregating the billing items", "computing the sales tax", "obtaining the customer address data", "producing the bill", and "dispatching the bill".

These business rules are candidates to be migrated to services. The second step is to assess the business value of these candidate services. The next step is to extract the existing legacy code which implements these business rules using the code stripping technique. The key idea in this technique is to identify the names of the essential data results and to trace how they are produced. This is achieved via an inverse data flow analysis. The data flow trace may pass through several methods or procedures in different classes or modules. For instance, in order to extract the bill customer service, several results including “billing items”, “sales tax” and “bill” should be traced back in the existing code. All the modules and classes identified should be combined to formulate the “bill customer” service. Once a code fragment has been identified as being a potential “bill customer” service, the next step is to extract it from the system in which it is embedded and to reassemble it as a separate module with its own interface. This is done by copying the impacted code units into a common framework and by placing all of the data objects they refer to into a common data interface. This entails generating a module which reads the input parameters “billing items”, “customer info” and “sale tax” from a WSDL input file and writes the result of “billing result” or error message into a WSDL output file.

3. The SOA Migration Framework

The SOA migration framework addresses the question of “what does the migration of legacy systems to SOA entail”. In [5] migration is defined as a modernization technique that moves the system to a new platform while retaining the original system data and functionality. According to Chikofsky [6], reengineering is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form. The commonalities among these two definitions are considerable. In practice, the notions of “legacy migration”, “integration” and “architectural recovery” which all deal with legacy applications are considered as approaches to reengineering. Following this line of thought, we consider the problem of migration of legacy systems to SOA as a reengineering problem.

According to [7] any type of reengineering can be comprised of three basic reengineering processes: 1) analysis of an existing system, 2) logical transformation, and 3) development of a new system. In the context of architectural recovery, a conceptual “horseshoe” model has been developed by the Software Engineering Institute, which distinguishes different levels of reengineering analysis and provides a foundation for transformations at each level, especially for transformations to the architectural level [7]. Given that migration is considered as some kind of reengineering process and that the horseshoe model is a generally accepted conceptual model for reengineering, we propose an extended form of the horseshoe model as a holistic model of the migration process.

Fig. 1 illustrates our proposed SOA migration framework (so called SOA-MF) for the migration process. Accordingly, a migration process follows a horseshoe model by first recovering the lost abstractions and eliciting the legacy fragments that are suitable for migration to SOA (left side), then altering and reshaping the legacy abstractions to service based abstractions (transformations in the middle area) and finally renovating the target system based on transformed abstractions as well as new requirements (right side). To adequately illustrate the notion of legacy migration we should recognize the corresponding key characterizing concepts. Migration processes are mostly comprised of three sub-processes including: reverse engineering transformation and forward engineering (thick arrows in Fig. 1). We argue that, the migration process is considered as transformation of the representations or artifacts (parallelograms in Fig. 1), which are carried out by means of a certain activity (rounded rectangles in Fig. 1). Activities can be supported by different types of knowledge as a resource or as the span of information that they should handle. Finally, the process of moving and mapping among the artifacts within the overall migration task, which graphically resembles a horseshoe, could be performed at different levels of abstraction ranging from “code-level” to “enterprise-level”.

3.1 Conceptual Elements of SOA-MF

In the following, we provide a systematic presentation of the main building blocks of the migration framework, so called conceptual elements. Below each conceptual element is presented along with its role in the migration process.

I. Process

According to Fayad, software processes define what needs to be done in a software development effort and how it is done [8]. Similarly, the migration process could be defined as the set of tasks that are carried out during migration. As mentioned, the migration process is also divided into three sub processes including reverse engineering, transformation and forward engineering. These sub-

processes are carried out through the four levels of abstraction, which will be discussed in Section 3.3. In the following, we discuss the role of these sub-processes within the whole migration process.

Reverse engineering is the process of analyzing the existing system to identify the system's structure, functionality and behavior and to create representation of the system in another form or at a higher level of abstraction [6]. It should be noted that, reverse engineering is the matter of examination and not a process of change. More precisely, "meaningful higher level abstractions" of the existing system are identified based on bodies of knowledge addressing the domain, technology, architecture, etc. From the migration point of view, the main goal of the reverse engineering process is to reach an understanding of the legacy system to the extent to identify the best candidates among the existing legacy elements for migration to SOA. Here, legacy elements are inherently the "meaningful higher level abstraction", recovered by means of reverse engineering techniques. In other words, the output of reverse engineering process is a number of legacy assets (in different levels of abstraction) which are extracted by means of the reverse engineering techniques and are suitable for the migration purpose.

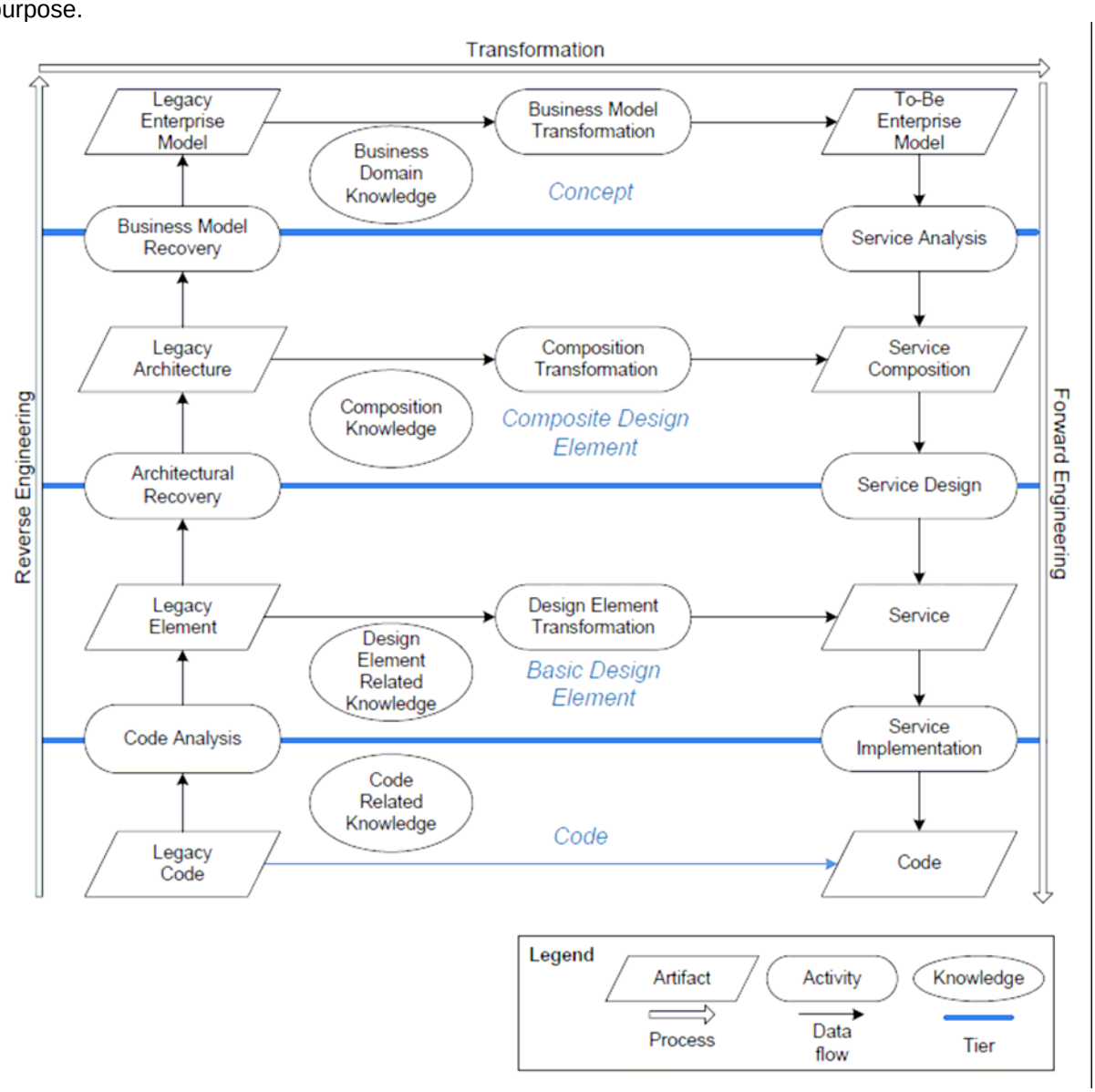

Figure 1: Overview of SOA Migration Framework

From the life-cycle coverage perspective, this process could start from existing implementation and continue with extracting the design entities, recovering the architecture and recapturing abstractions in requirements or business models.

Transformation is the process of restructuring one representation form to another at the same level of abstraction [6]. This transformation could be in the form of reshaping design elements, restructuring the architecture and/or altering business models and business strategies. It should be noted that, each

of these forms of transformation belongs to a specific level of abstraction. In other words, based on the level of abstraction of the initial and the target representations, different types of transformations are carried out. From the SOA migration perspective, transformation process embraces migration of legacy assets to service based assets. This process is performed through a set of activities associated to different levels of abstraction. Consequently, transformation in a particular migration approach could be performed within just one level or across number of them. In the example presented in Section 2, Sneed's approach just includes transformation of modules of legacy code to service implementation, while SMART encompasses reconstruction of the business model and architecture as well. Section 3.2 provides more description of different types of transformation activities. It is worth mentioning that transformation does not necessarily mean modification because of new requirements. Based on [6], transformation is altering "how" without altering "what".

During forward engineering, the subject system is renovated based on the new requirements and goals offered by the target service based environment as well as the target artifacts produced during the transformation process. Specifically, by considering the new requirements, goals, etc., and also the business model, service composition and/or services produced during the transformation process the service-based system is implemented in a top down manner.

II. Artifact

According to Conradi et al. [9], artifacts are the products of a process. Here, an artifact represents any product or "raw material" (i.e. models, architecture, piece of code) extracted, transformed or developed during each of reverse engineering, transformation or forward engineering processes.

III. Activity

An activity is an atomic or composite production step of a process which aims at generating or modifying a given set of artifacts [9]. Activities indicate regarding steps of what must be done during each of reverse engineering, transformation and forward engineering processes. Each of these activities is associated to a specific stage in the migration lifecycle.

IV. Knowledge

The Oxford Dictionary defines knowledge as "specific information; facts or intelligence about something". Here, we define knowledge as a term used to describe the whole spectrum of content for the following concepts concerning the migration process: data, models, procedures, techniques, principles, and context. Basically, these concepts are the set of information about software systems and business domain which shape the migration process given that they provide inputs to the migration activities. Different approaches are distinguished based on types of knowledge they exploit during the migration process. In [7] the set of levels of information about software system from the source code level to the architectural level is proposed. We follow the same view and classify the bodies of knowledge exploited within migration based on their associated level of abstraction. More precisely, where knowledge originates from (which level of abstraction), indicates the type of knowledge. As a consequence, the corresponding categories of knowledge are as follows: code-related knowledge, design element-related knowledge, composition knowledge and business domain knowledge. As an example, code grammars and models are categorized as code-related knowledge and are used within the reverse engineering sub-process. Cohesion is considered as a design element-related knowledge since it addresses a principle about a single element (i.e. module, component or service). Architectural patterns and styles are of type composition knowledge and finally business rules, risks, benefits and plans could be categorized as a business domain type of knowledge.

3.2 SOA-MF Description: Process Phases

Migration processes can be built in many different ways, i.e. it is not possible to identify one universal migration process. However, a basic general migration process which can constitute the skeleton of migration process at its most complete form is feasible to be defined. Lets recall that SOA-MF is devised based on the same scheme (skeleton of a complete migration process), that migration approaches cover a portion (or all) of.

So far, we have discussed our definition of migration process and the conceptual elements providing basis to describe the process. Now it is time to describe SOA-MF addressing the migration of legacy systems to SOA. The framework illustrates the migration process together with details of the artifacts included, activities carried out and types of knowledge exploited within each of migration sub-processes. The graphical representations of the conceptual elements are depicted in Fig. 1. The sub-processes, activities, artifacts and knowledge elements are respectively depicted by thick arrows,

rounded rectangles and parallelograms. In this section, the role of each activity in the whole migration process and the associated input and output artifacts as well as the knowledge exploited by the activity is described. The conceptual elements comprising the framework are written in italic.

I. Reverse Engineering

In its most basic form, reverse engineering starts from analyzing the legacy code within the code analysis activity. This activity aims at extraction of the legacy elements identified as candidates for transformation to services. Code analysis techniques such as graph-based analysis, lexical analysis, code querying, etc., are considered as instances of the code analysis activity. In the flight booking migration example, code stripping is considered as a code analysis activity. Regarding this activity, the input artifact is the legacy code while the output consists of set of legacy elements (which could be in the form of components, modules, segments of code, etc.). The extraction of legacy elements from code is influenced by involvement of code related knowledge (such as code grammar and model) as well as bodies of knowledge addressing higher level concepts (such as business domain knowledge). For instance, in the example presented in Section 2.2, code model (i.e. code related knowledge) and business rules (i.e. business domain knowledge) are considered as inputs to the code stripping activity.

So far, within the reverse engineering sub process, the extracted legacy elements are inherently design entities recaptured by means of reverse engineering techniques. However, we could go one step further and recapture the meaningful compositions of these legacy elements within the architectural recovery activity. Here, the composition knowledge such as architectural patterns and architectural styles are involved in identification of the architectural elements and their associated relationships.

Finally, the legacy enterprise model is extracted during the business model recovery activity. The inputs to this activity are the legacy architecture as well as the existing business domain knowledge such as business rules, business processes, etc.

II. Transformation

As mentioned, transformation encompasses process of restructuring one representation form to another at the same level of abstraction. The activities of design element transformation, composition transformation and business model transformation, respectively, realize the tasks of reshaping design elements, restructuring the architecture and altering business models and business strategies. The bridge part of the SOA-MF represents these transformations.

Design element transformation activity is typically performed to move the encapsulation of the legacy elements (extracted during the reverse engineering process) to services. Most of the wrapping techniques fall in this category of transformations. The input artifact to this activity is the legacy element (i.e. module, component or segment of a code) and the output artifact is basically a service. Types of knowledge which are inputs to design element transformation are: code related and design element knowledge.

Composition transformation activity embodies transformation of the legacy architecture (input artifact) to service compositions (output artifact) in terms of changing the allocation of functionality, their topology, etc. In other words, components and connectors are transformed to a service composition embracing services and relationships among them. Pattern based architectural transformation techniques fall in this category of transformations. Commonly, this activity exploits composition knowledge and design element knowledge as inputs to perform the transformation. For instance, architectural patterns, service composition patterns and service inventory patterns (i.e. composition knowledge) are used within the composition transformation activity.

During business model transformation activity the existing business model is transformed to a to-be business model based on new requirements as well as opportunities offered by service based systems. Here, existing business rules, business processes and strategies which are partially embedded in the legacy enterprise model are transformed to new ones to form the basis for development of service based system. The input artifact to this activity is legacy enterprise model, whereas the to-be enterprise model forms the output. The business model transformation activity is assisted by the business domain knowledge such as business rules, risks, benefits and plans.

III. Forward Engineering

In its most complete form, the forward engineering process starts from the to-be enterprise model. During service analysis, based on the to-be enterprise model a set of candidate service compositions which conceptualize the business processes are identified. Afterwards, the candidate service compositions are consolidated with service compositions identified during composition transformation

activity. Here, business domain knowledge as well as composition knowledge facilitate the service composition identification. This activity is succeeded by service design during which the renovated services are designed based on the consolidated candidate service compositions. Similar to service compositions, candidate services are merged with the services identified during design element transformation activity (of transformation process). Finally, during service implementation the service design is transformed to code.

3.3 Tiers

Transformation process embraces reshaping of an artifact of the existing legacy system to another artifact in the service oriented system. Considering four levels of abstraction including code, basic design elements, composite design element and concept, we argue that usually the as-is artifact (in reverse engineering process), the to-be artifact (in the forward engineering process) and their associated transformation activity all reside in the same level of abstraction. In that case, if we consider the as-is and to-be artifacts as well as the transformation among them as a tier, we could characterize and classify a migration approach based on the tiers supported. Fig. 1 depicts the tiers distinguished from each other by solid lines. It should be noted that, in a sample migration process, different set of tiers could exist, which are not necessarily adjacent. Consider a migration method which covers the concept and design element tiers. This implies that the business model transformation and design element transformation activities are included in transformation process whereas no composition transformation is carried out. From another point of view, a transformation in higher level of abstraction may not entail the transformation in lower levels.

4. On the role of SOA-MF

As mentioned previously, the main goal of the SOA-MF is understanding through classification and comparison of existing SOA migration methods. We argue that, SOA-MF is an intuitive graphical representation, which provides pieces of information to illustrate and characterize each existing migration process. More precisely, each migration process could be described based on the processes it supports, artifacts included, activities carried out, types of knowledge exploited and finally tiers they reside in. In other words, if we consider the SOA-MF model as a diagram, each migration process constitutes a portion of this diagram including the covered conceptual elements. Existing migration approaches use different terms and expressions for inherently similar tasks and concepts regarding migration. A general understanding of the migration methods could be reached by mapping and positioning the methods and their associated tasks and artifacts into a common framework. In the same vein, the SOA-MF model facilitates understanding and classification of existing migration methods and possibly provides the basis for analyzing their limitations and pitfalls. This is realized by identifying each method's associated portion of the diagram and comparing them based on their supported conceptual elements and their position related to the SOA-MF model (main diagram). To identify and map the framework elements in a possible migration process, Table 2 provides a number of questions that facilitates this mapping. We have studied a number of SOA migration approaches, and mapped and positioned them on SOA-MF. Two of these approaches, introduced in Section 2, are discussed here to clarify how a SOA migration approach can be mapped on SOA-MF. This implies that the processes, activities, artifacts and knowledge elements residing in each approach are mapped to an associated conceptual element in the SOA-MF. The position of each method is presented by representation of the related projection of the SOA-MF.

	Questions
Reverse Engineering	Does the migration method include activities related to extraction of abstractions or legacy elements as candidate for abstraction At what stage does the reengineering process start If reengineering is started from code what type of code analysis technique is used Is any kind of architectural recovery techniques included in the method Are business processes and business rules (partially) extracted from legacy system
Transformation	Does the method include techniques for transformation of legacy elements to services What artifact from legacy system is transformed to service based system Is the composition of legacy elements transformed to a service composition Are the existing business processes business rules and strategies altered in order to meet the new requirements and goals
Forward Engineering	Does the method include activities related to service development At what stage does the forward engineering process start

Table 2: Questions for Mapping a Migration Process on SOA-MF

4.1 Mapping of SMART Approach on SOA-MF

The bodies of information targeting the existing legacy system is extracted within the "describe candidate services" and "describe existing capabilities" activities. We classify this set of information (i.e. the stakeholders, business processes, goals, type of platform, data on the existing components, code, etc.) as existing enterprise model. The transformation sub-process embraces development of the migration strategy for transforming the existing capabilities to target SOA environment. The migration strategy is developed within "gap analysis" and "developed migration strategy" activities and are both categorized as business domain transformation activity (see Table. 1). The output of the transformation activity, categorized as to-be enterprise model, encompasses standards and guidelines for the service implementation, information on target environment, interaction of services on SOA environment, QoS expectations, etc. In addition, the business goals including costs, efforts and risks of migration that are categorized as business domain knowledge derive the development of migration strategy. It should be noted that, although this method also considers the information regarding to existing legacy code, components and architecture, we assign it to the concept tier since these bodies of information are elicited using high level analysis techniques such as interviews with stakeholders. Besides, the output of this method is an enterprise level plan for migration which is also dedicated to the concept tier.

4.2 Mapping of Sneed's Approach on SOA-MF

The first activity in this approach is identification of business rules that are considered as candidates for migration to web services. This is considered as codifying the knowledge regarding existing business processes, categorized as business domain knowledge. Reverse engineering sub-process starts from extraction of segments of code realizing business rules with good reuse potential. This is realized by the code stripping techniques that we categorize as a code analysis activity. The inputs of this activity are business rules (i.e. business domain knowledge) and the existing legacy code. Besides, the code grammars and code models that we categorize as code level knowledge facilitate the extraction of business rules from code. The transformation sub-process encompasses transformation of extracted modules to web services by means of wrapping techniques described in Section 2.2. Since the transformation encompasses wrapping a basic design element such as modules and altering them to web services, we categorize it as basic design element transformation activity. The forward engineering sub-process enables the implementation of the business processes, which is categorized as service implementation activity. Accordingly, this method is dedicated to basic design element tier.

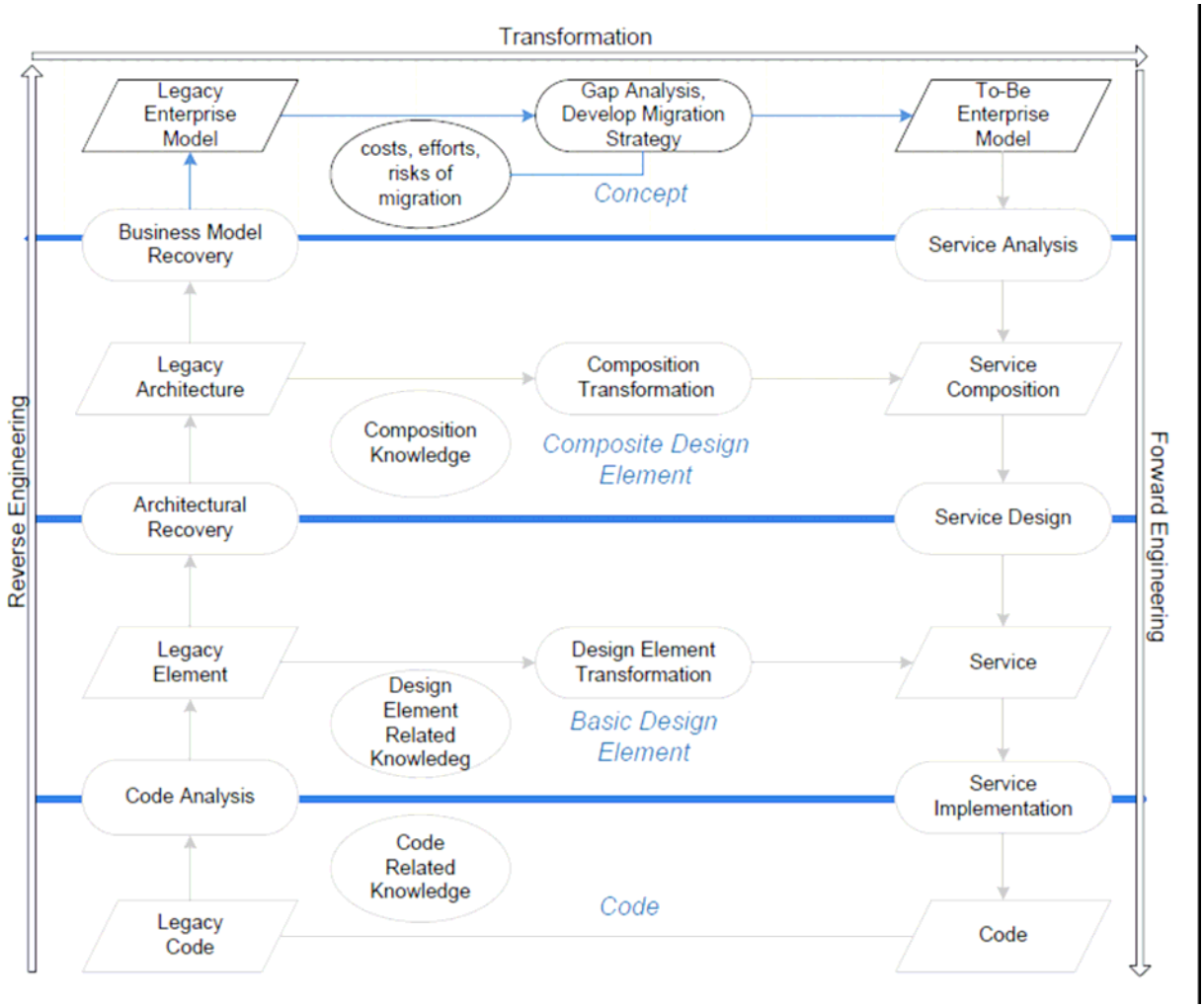

Figure 2: Mapping of SMART on SOA-MF

4.3 What the Mappings Imply

The mappings of the two example approaches already reflect the commonalties and differences among them. In this section, we provide the answer to the question of what do these two mappings imply and how they facilitate the understanding and comparison of the approaches.

Fig. 2, which illustrates the mapping of SMART on SOA-MF, reflects the following features of SMART: from lifecycle-coverage perspective, the reverse engineering and the forward engineering sub-processes are not covered. As a result, recapturing the abstractions of existing legacy system are not addressed. In addition, development of the service based system is not covered. Transformation is carried out at concept level and embraces altering legacy enterprise model of the existing system to the to-be enterprise model using business domain level knowledge.

The following features of Sneed's migration approach can be extracted from the associated mapping on SOA-MF. The horseshoe like representation of this approach on SOA-MF illustrates that all three sub-processes of reveres engineering, transformation and forward engineering are carried out. The transformation occurs at basic design element level and migration is limited to altering modules to services, however, business domain knowledge facilitates the migration.

To sum up, SMART provides high level solutions for migration problem, while ignoring the technical details of legacy element extraction and service development. Whereas, Sneed's approach presents a migration process at lower level through focusing on technical details.

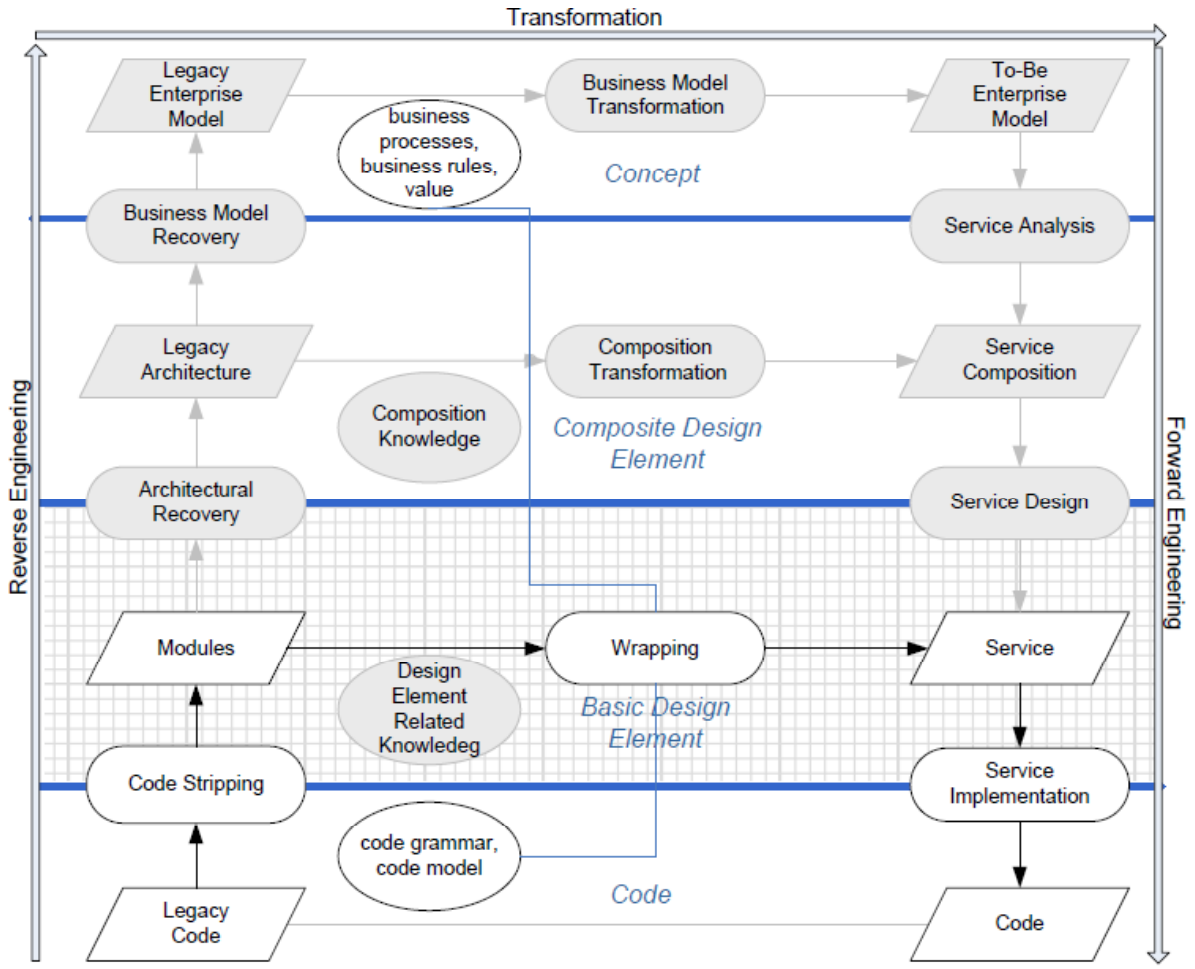

Figure 3: Mapping of Sneed's approach on SOA-MF

5. Conclusion

It is hard to understand and classify existing approaches in an emerging and still fuzzy research field like SOA migration. In the past few years, several SOA migration approaches have been introduced, each focusing on a specific perspective on SOA migration, and using own concepts (activities, artifacts, etc.) to represent migration.

This paper has presented a SOA migration framework (SOA-MF). SOA-MF facilitates to characterize and isolate the properties of migration approaches in terms of processes it supports, artifacts included, activities carried out, and types of knowledge exploited. A unified representation of different approaches is achieved by mapping them on SOA-MF, which also provides the basis for their comparison and analysis.

During experimentation with earlier versions of SOA-MF, we observed that the notion of tier plays an important role in positioning and classifying the various migration approaches. The tiers of SOA-MF covered by a specific SOA migration approach can explain the following aspects: the associated level of abstraction in which the transformation occurs and the transformations that entail lower level ones. A relevant classification of existing approaches can be achieved by considering tiers they cover in SOA-MF. For instance, SMART and Umar's [4] approaches are dedicated to the concept tier category (in which just the concept tier is covered); while Sneed's method is regarded as a basic design element tier approach; and the process presented in [10] is dedicated to the category of migration processes covering all tiers.

As our future work, we will use SOA-MF in industrial organizations to elicit how industry performs SOA migration in practice and see if we can recognize generic patterns. Moreover, to evaluate and refine SOA-MF, we are also bringing it to perform a systematic literature review on existing SOA migration processes.

Acknowledgement

This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research and Development (Jacquard) program on Software Engineering Research via contract 638.001.206 SAPIENSA: Service-enAbling PreexIsting ENterprISe Assets.

References

1. Aversano, L., Canfora, G., Cimitile, A., de Lucia, A.: Migrating legacy systems to the web: an experience report. In: CSMR ’01: Proc. of the 5th European Conference on Software Maintenance and Reengineering, Washington, DC, USA, IEEE Computer Society (2001) 148
2. Sneed, H.M.: Integrating legacy software into a service oriented architecture. In: CSMR ’06: Proc. of the Conference on Software Maintenance and Reengineering, Washington, DC, USA, I EEE Computer Society (2006) 3–14
3. Lewis, G., Morris, E., Smith, D., O’Brien, L.: Service-oriented migration and reuse technique (smart). In: STEP ’05: Proc. of the 13th IEEE International Workshop on Software Technology and Engineering Practice, Washington, DC, USA, IEEE Computer Society (2005) 222–229
4. Umar, A., Zordan, A.: Reengineering for service oriented architectures: A strategic decision model for integration versus migration. Journal of Systems and Software 82(3) (2009) 448 – 462
5. Bisbal, J., Lawless, D., Wu, B., Grimson, J.: Legacy information systems: Issues and directions. IEEE Software 16 (1999) 103–111
6. Chikofsky, E.J., II, J.H.C.: Reverse engineering and design recovery: A taxonomy. IEEE Software 7(1) (1990) 13–17
7. Kazman, R., Woods, S.G., Carri`ere, S.J.: Requirements for integrating software architecture and reengineering models: CORUM II. (1998) 154
8. Fayad, M.E.: Software development process: a necessary evil. Commun. ACM 40(9) (1997) 101–103
9. Conradi, R., Fernstr¨om, C., Fuggetta, A.: A conceptual framework for evolving software processes. SIGSOFT Softw. Eng. Notes 18(4) (1993) 26–35
10. Andreas Winter, J.Z.: Model-based migration to service-oriented architecture. In: the International Workshop on SOA Maintenance Evolution (SOAM 2007). (2007)
11. Razavian, M., Lago, P.: Towards a conceptual framework for legacy to soa migration. In: Fifth International Workshop on Engineering Service-Oriented Applications (WESOA'09). (2009)